REMARKS/ARGUMENTS

This Amendment and the following remarks are intended to fully respond to the Office Action mailed September 22, 2005. In that Office Action, claims 1-34 and 38-58 were examined, and all claims were rejected. More specifically, claims 1-34 and 38-58 stand rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention; claims 1-4, 6-7, 9-13, 15, 17-23, 25-34, 38-41, 43-44, 46-50, 52 and 54-58 stand rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,189,142 to Johnston et al. (hereinafter, "Johnston"); and claims 5, 8, 14, 16, 24, 42, 45, 51 and 53 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Johnston in view of U.S. Patent No. 6,349,406 to Levine et al. (hereinafter, "Levine"). Reconsideration of these objections and rejections, as they might apply to the original and amended claims in view of these remarks, is respectfully requested.

In this Response, claims 1, 19, 26, 31, 38, and 55 have been amended; no new claims have been added; and no claims have been newly canceled. Claims 35-37 remain canceled. The Applicant respectfully points out that claims 1, 19, 26, 31, 38, and 55 were amended only for clarification purposes.

Claim Rejections – 35 U.S.C. § 112

Claims 1-34 and 38-58 stand rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which the Applicant regards as the invention. Determining whether a claim is indefinite "requires an analysis of whether those persons skilled in the art would understand the bounds of the claim when read in light of the specification." Credle v. Bond, 25 F.3d 1566, 1576 (Fed. Cir. 1994) (emphasis added). The degree of precision necessary is a "function of the nature of the subject matter" at issue. Miles Laboratories, Inc. v. Shandon, Inc., 997 F.2d 870, 875 (Fed. Cir. 1993).

The Examiner rejected claims 1-34 and 38-58 for indefiniteness, arguing that "wherein the history operator is capable of being referenced directly by the source code" renders the claims "indefinite because the source code can not directly *reference* the history operator since source code is merely human-readable program statements written by a programmer in a high-level or assembly language that are not directly readable by a computer." Examiner's Response to Amendments, Office Action, 9/22/05, pp. 2-3 (emphasis in original). The Examiner then explains: "Source code needs to be compiled into object code before it can be executed by a computer. Object code can directly *reference* the history operator." Id. at 3 (emphasis in original). Further, the Examiner changes the wording of this limitation to read, "wherein the history operator is capable of being referenced directly by the second object code." Id. at 3 (emphasis added).

In the interest of clarifying the elements containing this language, the Applicant has herein amended the relevant claim language. For example, claim 1 now provides that "the history operator is capable of being referenced in the source code." It is well known in the art that a programmer, or computer developer, can reference an executable command in the source code. Thus, according to an embodiment of the present invention, the programmer would reference the history operator in the source code. This is the only way to create the reference in the object code. Thus, when "read in light of the specification," see Credle, a person of ordinary skill in the art would understand that this invention provides for the history operator to be referenced in the source code. The specification in the pending application expressly provides: "The source code contains a history operand, instances of the history operand, and the history operator." Application, p. 3 (emphasis added). See also id. at p. 2 (describing that "source code . . . contains the program-history construct"); Abstract, p. 35 (stating, "The source code contains . . . the history operator."). Further, FIG. 1 demonstrably shows the history operator as being contained within the source code and thus capable of being referenced in it. Figure 4 also explicitly shows the history operator, as shown in the particular embodiment of FIG. 4 as the "average" function, as being referenced directly by the code fragment 450. The code fragment 450 is contained within code segment 400 which, in turn, is contained within source code 205. See also FIGS. 5-12 (showing history operator as being ultimately referenced in the source code). Further, the application's "Summary of the Invention" provides that by use of the history operand and history operator, "the programmer is freed from writing tedious and error-prone bookkeeping code that requires variables and saves or computes history data into those variables." The history operand and history operator are thus substituted for traditional source code bookkeeping functionality. A program history provides functionality by exposing the history of values of a variable to a programmer for manipulation directly or aggregately, e.g., average, sum, etc.

A close reading of the specification indicates the Examiner's interpretation that the "history operator is capable of being referenced directly by the second object code" is unsupported by the plain language and Figures of the specification. The specification describes the inter-workings of the source code with the object code, in which it is explained, according to one embodiment of the invention, that "[t]he presence of the history operator [in the source code] directs the translator to generate object code that will perform a function on that saved history data." Id. From this statement, it is evident that the history operator ultimately directs the "generat[ion]" of the object code according to this embodiment of the invention. Similarly, both FIG. 13 and the description of this Figure provide for the step of "recogniz[ing] a history operator" *followed* by the step of "generat[ing] object code that will perform the history operator on the history." Further, according to certain embodiments of the present invention, the "history-processing program does not create object code, but instead saves history data." If no object code is created in the first instance, the Examiner's interpretation that the history operator is only referenced by object code is illogical and inconsistent with the plain language of the specification.

In stating that the history operator may be "referenced" in the source code, it should be noted that program histories are not active variables but are higher-level language features used to facilitate programming by providing implicit computation. Regardless of whether the computation is implicit, however, the history operator may still be referenced in the source code as a special construct which is then capable of recognition by the compiler or interpreter of the source code. See, e.g., Application at 19, lines 17-19 ("When programmers want to refer to the history of a control construct, they label the construct." (emphasis added)). Accordingly, history operators and operands can be substituted for traditional bookkeeping source code. In reciting an interpretation "wherein the history operator is capable of being referenced directly by the second object code," id. at 3 (emphasis added), the Examiner effectively changes the scope and subject matter of the invention itself. The Examiner's "interpretation" amounts to a sua sponte amendment to the Applicant's claims. With all due respect, the Applicant cannot agree with this interpretation.

Based on the current clarification amendments to the relevant claim language, the Applicant respectfully believes the Examiner's § 112 rejections are rendered moot. Accordingly, taking into consideration these amendments, and because a person skilled in the art would understand the bounds of claims 1-34 and 38-58 in light of the extensive disclosure provided in the specification portion of this application, the Applicant respectfully requests that the Examiner reconsider the 35 U.S.C. § 112, second paragraph, rejections of claims 1-34 and 38-58.

Claim Rejections – 35 U.S.C. § 102

Claims 1-4, 6, 7, 9-13, 15, 17-23, 25-34, 38-41, 43-44, 46-50, 52 and 54-58 stand rejected under 35 U.S.C. § 102(e) as being anticipated by Johnston. A prima facie case of anticipation can be met only where the reference teaches each and every aspect of the claimed invention. See MPEP §§ 706.02 & 2136. Applicant respectfully traverses the Examiner's rejections under 35 U.S.C. § 102(e) on the grounds that Johnston does not disclose, either explicitly or implicitly, each and every limitation of the pending claims. See Verdegaal Bros. v. Union Oil Co. of Cal., 814 F.2d 628 (Fed. Cir. 1987).

Embodiments of the present invention, as defined in the claims, generally relate to a program history. A program history is a programming-language construct. The program history captures data regarding the state of the program as the program itself executes and makes this data available to functions called history operators. According to embodiments of the present invention, the source code contains the program-history construct. Thus, contained within the source code are a history operand, instances of the history operand, and the history operator. History operators and operands may be used to optimize and condense source code by substituting for traditional source code bookkeeping functionality. Examples of history operators and history operands, and of source code optimizations enabled therewith, are shown in FIGS. 4-12 of the pending application. A history operand in source code represents a sequence of data associated with the history of an operand instance. A history operator in source code is a shorthand representation of a function that object code will perform on the history data represented by the history operand. In other words, history operators and operands are special, syntactical constructs that are inserted by the programmer and recognized by the compiler or interpreter of the source code that then trigger the compiler or interpreter to generate a particular object code. History operators and operands are unique in that their values can both be accessed through the source code and can be used to determine code flow.

Johnston, on the other hand, relates to "providing runtime performance analysis in a visual programming environment." Johnston provides for the gathering of performance tracing information when performance benchmarks are executed against the visually-created program being analyzed. The performance benchmarking process comprises "adding code hooks in appropriate locations within the code that implements the visual program." Johnston, col. 9, lines 43-51. Adding hooks may be accomplished by "adding a small piece of code in the appropriate place for each element of interest." " Id. at col. 9, lines 54-59. The code hooks in Johnston are specifically used to "log[] the fact that the invoking element has executed; causes recording of execution timing information; and for attribute-to-attribute connections, logs the amount of data being moved." Id. at col. 9, lines 64-67 (emphasis added). The code hooks in Johnston are set up when the application is originally developed and, thus, are limited in scope and can only be selected (turned on or off). The history operators of the present invention, on the other hand, are inserted in the source code allowing greater control by the programmer. Johnston's data is limited to what the developing programmer originally set up, while the history operator enables the data to be manipulated by the programmer using the history operator functions (e.g., determining minimum, maximum, average).

With respect to claim 1, Johnston thus fails to teach "recognizing a history operator and a history operand in the source code;" "generating first object code that, when executed, saves a data history associated with an instance of the history operand;" "generating second object code that, when executed, performs the history operator on the data history," or "wherein the history operator is capable of being referenced in the source code." While Johnston discloses the use of "code hooks" for runtime performance analysis, "code hooks" are not "history operators" or "history operands." History operators and history operands are programming language statements, or portions thereof, which capture aspects of past program states implicitly. Code hooks, on the other hand, merely run within the program to monitor code flow and execution without being used themselves to alter code flow. See Microsoft Computer Dict. 256 (defining "hook" as "[a] location in a routine or program in which the programmer can connect or insert other routines for the purpose of debugging or enhancing functionality."). Unlike history operands and history operators, Johnston's code hooks serve only to log and record data and thus do not return any values. Code hooks and history operators and operands are thus entirely distinct concepts.

Further, the distinct nature of Johnston's use of code hooks and the present invention's history operators and operands can be seen by examining the purposes of these inventions. First, Johnston's code hooks are used to perform bookkeeping functions, i.e., "to count the number of times a particular visual programming element is executed," according to one embodiment.

Johnston, col. 3, lines 46-48. The code hooks of Johnston are used by "adding a small piece of code in the appropriate place for each element of interest." Col. 9, lines 54-55. The present invention, on the other hand, teaches away from the use of "tedious" bookkeeping programming by using program histories. In fact, it is the use of program histories that enables programmers to access past program state without having to use tedious bookkeeping code, such as the code hooks defined in Johnston.

Second, Johnston's specific use of a runtime performance analysis in a "visual" programming environment necessarily limits its applicability to the invention here which is code-based in accordance with certain embodiments of the invention. Indeed, Johnston draws sharp distinctions between component-based visual programs and conventional programs:

> [C]omponent-based visual programs are different from conventional programs in several important ways. Because of these differences, known performance monitoring and analysis techniques cannot be applied to this new programming domain in a way that provides meaningful results. A first difference between the visual programs and conventional programs relates to the types of building blocks used by a programmer in creating a program, and a second difference lies in the source of those building blocks. Visual programs are graphically constructed primarily from pre-built components . . . [a] conventional programmer, on the other hand, typically writes program code himself using textual source-code statements of a programming language . . . .

Johnston, col. 2, lines 11-34.

Thus, in relating to a "visual programming environment," Johnston expressly teaches away from the present invention. For example, in accordance with an embodiment of the present invention, the invention provides a "series of code statements that prints the number of warning messages issued by reporting the number of calls to the function 'warning.'" Application, p. 18, lines 5-6 (emphasis added). In contrast, Johnston states, "[I]t is not useful to calculate performance data for a visual program . . . based on how many times a function was called . . . or other source-code-based approaches currently used for conventional programs." Johnston, col. 6, lines 36-41 (emphasis added). Johnston and the present invention thus emanate from conceptually distinct and different fields of art.

For these reasons, the Applicant respectfully maintains that Johnston cannot anticipate the "history operands" and "history operators" as recited in claim 1 of the present invention. Reconsideration is thus respectfully requested. Given that claims 2-4, 6-7, 9-13, 15, and 17-18 depend directly or indirectly from allowable base claim 1, these claims are also believed to be patentable over Johnston.

Similarly, with respect to claims 19, 26, 31, 38, and 55, Johnston fails to teach a "history operand" or "history operator." Accordingly, as a matter of law and as explained above, Johnston cannot anticipate claims 19, 26, 31, 38, and 55. Given that claims 20-23, 25, 27-30, 32-34, 39-41, 43-44, 46-50, 52, 54, and 56-58 depend directly or indirectly from allowable base claims 19, 26, 31, 38 and 55, these dependent claims are also believed to be patentable over Johnston.

Accordingly, the Applicant respectfully requests that, at the Examiner's earliest convenience, a notice of allowance be issued for all claims.

Claim Rejections – 35 U.S.C. § 103

Claims 5, 8, 14, 16, 24, 42, 45, 51 and 53 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Johnston in view of Levine. Applicant respectfully traverses the § 103(a) rejections because the Examiner has failed to state a prima facie case of obviousness. A prima facie case of obviousness can be established only when all of the following requirements are satisfied: (1) the reference or combination of references must teach or suggest all of the claim limitations; (2) there must be some suggestion or motivation in the references themselves to combine the references; and (3) there must be a reasonable expectation of success. See MPEP §§ 706.02(j) & 2143. Thus, the combination of references cited by the Examiner must teach or suggest every limitation of the claimed invention. CFMT, Inc. v. YieldUp Int'l Corp., 349 F.3d 1333, 1342 (Fed. Cir. 2003); see also MPEP § 2143.03.

Each of the claims rejected under § 103(a) is a dependent claim. Claims 5, 8, 14, and 16 are dependent upon independent claim 1; claim 24 is dependent upon independent claim 19; and claims 42, 45, 51, and 53 are dependent upon independent claim 38. As argued above, Applicant respectfully maintains that claims 1, 19, and 38 are allowable base claims. The Examiner has made no § 103(a) rejections of claims 1, 19 or 38 but, instead, has limited the § 103(a) rejections to claims 5, 8, 14, 16, 24, 42, 45, 51, and 53. Because Applicant respectfully maintains that claims 1, 19, and 38 are allowable claims, claims 5, 8, 14, 16, 24, 42, 45, 51, and 53 are also believed to be patentable over Johnston in view of Levine because these claims depend from the allowable independent claims 1, 19, and 38.

Accordingly, the Applicant respectfully believes the Examiner's § 103(a) rejections are rendered moot in light of the preceding arguments in favor of patentability. The failure, if any, of this Amendment and Response to directly address a § 103(a) argument raised by the Examiner should not be interpreted as indicative of the Applicant's belief that such argument has any merit. Thus, the preceding arguments in favor of patentability are advanced without prejudice to other bases of patentability.

Conclusion

This Amendment and Response fully responds to the Office Action mailed on September 22, 2005. It is recognized that the Office Action may contain arguments and rejections that are not directly addressed by this Amendment and Response due to the fact that they are rendered moot in light of the preceding arguments in favor of patentability. Hence, the failure, if any, of this Amendment and Response to directly address an argument raised by the Examiner should not be interpreted as reflecting the Applicant's belief that such argument has merit. Furthermore, the claims of the present application may include other elements, not discussed in this Amendment and Response, which are not shown, taught, or otherwise suggested by the art of record. Accordingly, the preceding arguments in favor of patentability are advanced without prejudice to other bases of patentability.

It is believed that no further fees are due with this Amendment and Response. However, the Commissioner is hereby authorized to charge any deficiencies or credit any overpayment with respect to this patent application to deposit account number 13-2725.

In light of the above remarks and amendments, it is believed that the application is now in condition for allowance and such action is respectfully requested. Should any additional issues need to be resolved, the Examiner is requested to telephone the undersigned to attempt to resolve those issues.

Respectfully submitted,

Dated: April 3, 2006

[illegible]

René A. Pereyra, #45,800
MERCHANT & GOULD P.C.
P.O. Box 2903
Minneapolis, MN 55402-0903
303.357.1648

27488
PATENT TRADEMARK OFFICE